Nokia Corporation Docket No.: NC39081
Harrington & Smith, LLP Docket No.: 883.0006.U1(US)
Application for United States Letters Patent by:
Yoshiya Hirase

DYNAMIC PROPERTY INHERITANCE ALGORITHM FOR SCHEDULING DYNAMIC CONFIGURABLE DEVICES

DYNAMIC PROPERTY INHERITANCE ALGORITHM FOR SCHEDULING DYNAMIC CONFIGURABLE DEVICES

CLAIM OF PRIORITY FROM A COPENDING U.S. PROVISIONAL PATENT APPLICATION:

This patent application claims priority under 35 U.S.C. 119(e) from Provisional Patent Application No.: 60/436,771, filed 12/26/2002, the content of which is incorporated by reference herein in its entirety.

TECHNICAL FIELD:

These teachings relate generally to computer operating systems and architectures, and more specifically relate to methods and apparatus that employ configurable hardware for implementing devices, such as handheld communicators and cellular telephones, which may be referred to as mobile terminals, and other types of user devices, such as personal digital assistants (PDAs).

BACKGROUND:

Configurable hardware has not yet been implemented into commercial mobile terminals due to a lack of maturity of the technology, but future generation mobile terminals and other products are expected to require this type of hardware architecture in order to reduce power consumption and extend their functionality to new and more demanding applications, such as multi-media applications.

The conventional approach to mobile terminal design is to employ a general purpose digital signal processor (DSP) and possibly a custom integrated circuit, such as an ASIC, for the desired application(s). However, this conventional approach is proving to be less than adequate as mobile terminal applications increase in complexity and processing requirements. This is true at least for the reasons that the power consumption can be increased to the point that the power dissipation within the device becomes an important issue, and a lack of flexibility can result in wasted resources if the overall architecture

must be designed to accommodate the most demanding applications.

SUMMARY OF THE PREFERRED EMBODIMENTS

The foregoing and other problems are overcome, and other advantages are realized, in accordance with the presently preferred embodiments of these teachings.

This invention provides a core algorithm for use in scheduling the timing of algorithm logic for Dynamic Configurable Hardware Logic (DCHL). The scheduling algorithm is referred to herein as "Priority Inheritance" (PI). The PI exploits the potential of DCHL to accelerate the execution of multi-media application software, as the logic utilization of DCHL is not optimum without the use of the PI in accordance with this invention. DCHL is expected to be used as a extensively adaptable hardware accelerator in mobile terminals. Multi-media applications within a host CPU will operate with appropriate algorithm logic within the DCHL to accelerate multi-media application execution, and to enhance the functionality of multi-media applications. Since applications are scheduled in accordance with such priorities, the PI is an important mechanism to achieve an optimal utilization of DCHL.

Disclosed is a device architecture for running applications. The device architecture includes an operating system (OS) having an OS scheduler, a Dynamic Configurable Hardware Logic (DCHL) layer comprised of a plurality of Logic Elements (LEs) and, interposed between the OS and the DCHL layer, a TiEred Multi-media Acceleration Scheduler (TEMAS) that cooperates with the OS scheduler for scheduling the LEs of the DCHL to execute applications.

In accordance with this invention, the scheduling uses inherited application priorities so that the algorithms begin to execute at the correct times, and without incurring any inefficient DCHL configuration costs.

In the preferred embodiment the TEMAS is constructed to contain a Tier-1 scheduler that communicates with the OS scheduler, and at least one Tier-2 scheduler interposed

between the Tier-1 scheduler and one DCHL configurable device.

Also disclosed is a method to execute applications in a device. The method includes providing an OS comprising an OS scheduler and a DCHL layer comprised of a plurality of LEs; interposing between the OS and the DCHL layer TEMAS and operating the TEMAS in cooperation with the OS scheduler for scheduling the LEs of the DCHL to execute applications in accordance with inherited application priorities.

Also disclosed is a wireless communications device, such as a cellular telephone, that includes an applications layer comprising a plurality of applications; a service layer comprising an OS having an OS scheduler; a hardware layer comprising DCHL comprised of a plurality of LEs and, interposed between the OS and the DCHL in the service layer and in a node layer, the TEMAS that cooperates with the OS scheduler for scheduling the LEs of the DCHL to execute the applications in accordance with inherited application priorities.

BRIEF DESCRIPTION OF THE DRAWINGS

The foregoing and other aspects of these teachings are made more evident in the following Detailed Description of the Preferred Embodiments, when read in conjunction with the attached Drawing Figures, wherein:

Fig. 1 is a diagram that is useful for introducing Dynamic Configurable Hardware Logic (DCHL) terminology;

Fig. 2 shows an example of the configuration of the DCHL Logic Elements (LEs);

Fig. 3 shows an assumed mobile terminal DCHL-based architecture;

Figs 4A and 4B, collectively referred to as Fig. 4, show the TEMAS, where Fig. 4A is a simplified block diagram of the TEMAS in the context of a multi-layered device architecture, and shows the TEMAS interposed between the application layer, the OS

and the DCHL hardware layer, and is illustrative of a run-time framework, and Fig. 4B is a diagram that depicts an implementation example of the TEMAS;

Fig. 5 shows the reconfiguration of the DCHL in response to configuration requests from the Tier-2 schedulers of Fig. 4, where the DCHL can accept more than one algorithm logic at the same time, and where a given algorithm logic can be configured into the DCHL while another one or ones are currently operating in the DCHL;

Fig. 6 is a diagram that provides an overview of the Priority Inheritance algorithm;

Fig.7 shows an example of application and algorithm logic execution where there is no Priority Inheritance algorithm in accordance with this invention;

Fig.8 shows an example of application and algorithm logic execution where there is the Priority Inheritance algorithm in accordance with this invention;

Fig. 9 shows an implementation examples of the Priority Inheritance algorithm in a DCHL-based mobile terminal;

Figs. 10, 11 and 12 are referred to when explaining conventional uses of Priority Inheritance; and

Fig. 13 illustrates the use of the Priority Inheritance algorithm of this invention in allocating optimal algorithm logics onto a DCHL resource at a correct time, and without incurring additional configuration overhead.

DETAILED DESCRIPTION OF THE PREFERRED EMBODIMENTS

One approach to overcoming the problems inherent in the use of generic DSPs and custom logic can be referred to as Dynamic Configurable Hardware Logic (DCHL). As can be seen in Fig. 1, the basic unit of the DCHL architecture is the Logic Element (LE) 10, a plurality of which are arranged in a context plane 12, or more simply a context. A

plurality of context planes 12 result in a multi-context 14. The LE 10 is a unit to be configured as an algorithm logic, and an algorithm logic is assumed to include a set of LEs 10. More than one context 12 can be included in a single device, and more than one algorithm logic can be configured into one context 12, and can operate simultaneously.

As is shown in Fig. 2, some LEs 10 can be configured partially (e.g., switched from algorithm logic-2 to algorithm logic-3) while other LEs 10 are operating (algorithm logic-1). One of the application logics (algorithm logic-2) is shown as being released after the configuration process.

A mobile terminal with DCHL 50, applications 30 and a generic operating system 40 (one not optimized for use with DCHL 50, such as Linux™) has the architecture shown in Fig. 3. For example, a host CPU includes a plurality of the applications 30, the generic operating system 30, and a DCHL scheduler, also referred to herein as the TEMAS 20. This architecture employs a runtime framework of a type shown in Fig. 4 to accelerate multi-media applications by the use of algorithm logics.

In accordance with an aspect of an invention described in commonly assigned U.S. Patent Application No.: 10/_____, filed on the same date as this Patent Application and entitled: Tiered Multi-media Acceleration Scheduler Architecture for Dynamic Configurable Devices, by Yoshiya Hirase (Attorney Docket Nos.: NC39080/883.0005.U1(US)), a multi-layered scheduler, such as a two-layered scheduler, is referred to as a TiEred Multi-media Acceleration Scheduler (hereinafter TEMAS) which deals with the DCHL layer as a generic system software model. Referring to Fig. 4A, four typical device layers are shown: the application layer 60 (containing three exemplary applications 30A, 30B, 30C), a service layer 80, a node layer 90 and the hardware layer 70 (the DCHL 50 layer). The Tier-1 Scheduler 82, containing a scheduling algorithm 82A, is shown resident at the service layer 80, while a plurality of Tier-2 Schedulers 92, 94, 96 are resident at the node layer 90, one for each node entity 90A. An OS Scheduler 45, part of the OS 40, is shown resident in the service layer 80 with the Tier-1 Scheduler 82. The OS Scheduler 45 manages all applications 30 that are ordinary applications, and multi-media applications that use the DCHL 50. The Tier-1

Scheduler 82 obtains scheduling information about multi-media applications from the OS Scheduler 45, via a Hook Module 47 (shown in Fig. 4B). The most important information is the scheduling order of applications 30 and the priorities (see (a) in Figs. 4B and 9). The scheduling order is used to decide when preloading is performed for the DCHL 50. The priority of the applications gives the actual priority of the algorithm logic to be configured into and executed on the DCHL 50 (see Fig. 5). Since the priority of the algorithm logic cannot be determined until an actual application is attached to it, this function of the Tier-1 Scheduler 82 is important (see scheduling of events (b) and (c) in Figs. 4B and 9). The Tier-1 Scheduler 82 also obtains communication overhead from the device driver, and determines the difference in timing for the DCHL hardware (see (d) in Fig. 4B), which aids in adjusting the scheduling timing. Additional algorithms can be extended using modules (see (e) in Fig. 4B). The Tier-2 Scheduler 92, 94, 96 receives configuration requests from the Tier-1 Scheduler 82 (see (b) in Figs. 4B and 9, as well as Fig. 5), and schedules the algorithm logic to be executed within the DCHL 50 (see (c) in Figs. 4B and 9). As may be apparent, the separation of the TEMAS 20 into the at least two layers (Tier-1 and Tier-2) allows any type of DCHL 50 to be compatible with the heuristic scheduler of the generic OS 40 architecture.

Fig. 5 shows the operation of the TEMAS 20 in response to configuration requests, that arrive during time steps, and the configuration and reconfiguration of the LEs of the DCHL 50 in accordance with various algorithm logics.

As was noted above, the OS scheduler manages all applications that are ordinary applications and multi-media applications using the DCHL 50. The priority of an application is collected from the OS Scheduler 45 by the Tier-1 Scheduler 82 via the above-referenced Hook Module 47 ((a) in Figs. 4B and 9). The Tier-1 Scheduler 82 determines which algorithm logic is appropriate to schedule and sends a scheduling event to the Tier-2 Scheduler 92 as (b) in Figs. 4B and 9. This triggers the Tier-2 Scheduler 92 to perform the actual scheduling of algorithm logic for DCHL as (c) in Figs. 4B and 9, and (d) in Fig. 9.

In accordance with an aspect of this invention, the Tier-2 Scheduler 92 uses the inherited

priority from the application 30, as obtained from the OS Scheduler 45, to determine the optimal scheduling of the algorithm logics in the DCHL 50. Under such a system, Fig. 8 shows the operation of the application scheduling mechanism, using the PI in accordance with this invention, as will be described in further detail below.

Discussing the PI now in further detail, note that so-called "Priority Inheritance" algorithms are generally known. However, conventional PIs differ from this invention in several respects. A first difference is that the PI avoids a deadlock condition when managing a software resource, which enables a higher priority task to obtain control of a required resource without deadlock when the resource is already claimed by a lower priority task. The locking task is executed with the inherited higher priority until the locked resource is released (see Fig. 10). This conventional use of the PI is thus more of an attempt to manage a shared resource.

A second difference relates to enhancing the performance of device drivers (see, for example, U.S. Patent No.: 5,469,571). In this conventional case the PI allows a kernel thread handling a device driver to serve hardware interrupts in accordance with the priority of the associated user thread by inheriting the priority from the user thread (see Figs. 11 and 12). The result is that an application having a higher priority can obtain data efficiently from the lower layer of device drivers. This use of PI is focused on the use of a decision as to what order the kernel would execute kernel threads. More particularly, and by example, if two user threads exist, and each uses a different device driver, kernel threads are created to handle the data received from the device drivers. The inherited priorities attached to kernel threads are used to decide in what order the kernel threads should be executed.

This also differs from the use of the PI by this invention, at least partly because the PI of this invention is directed towards allocating optimal algorithm logics onto a DCHL 50 resource at a correct time, and without incurring additional configuration overhead (see Fig. 13). In the example case where two user threads exist, and use algorithm logics already allocated in the DCHL 50, a request for allocation of a third user thread (user thread-3) may be rejected based on the inherited priority of the third user thread.

Fig. 7 is an example of how problems can occur without the use of the PI of this invention. Note that application-1 must wait (execution is delayed) from time (a) to time (d) because the additional configuration (b) that is required after the needless configuration of (4). Further, algorithm-3 misses execution at correct time (c).

More specifically, at first application-1 with priority 1 uses Algorithm Logic-1 after configuring the DCHL 50 at times (1) and (2). Next, Algorithm Logic-2 starts by requesting from application-2 with priority 2 at time (3). At this point, the DCHL 50 has two algorithm logics. Then, when application-2 with priority 2 requests additional Algorithm Logic-3 at time (4), the first loaded Algorithm Logic-1 is released to configure Algorithm Logic-3 due to the limited amount of DCHL 50 capacity. Application-1 must then wait from (a) to (d) because of the additional configuration (b) that is required after the useless configuration that was performed at time (4). And Algorithm-3 misses its execution at the correct time.

Fig 8 shows the benefit of the PI mechanism of this invention. Since all of the priorities associated with the applications 30 are inherited by the attached algorithm logics, the scheduling for the DCHL 50 can be performed optimally. All of the algorithms begin to operate at the correct times, without incurring any inefficient configuration costs.

The foregoing description has provided by way of exemplary and non-limiting examples a full and informative description of the best method and apparatus presently contemplated by the inventor for carrying out the invention. However, various modifications and adaptations may become apparent to those skilled in the relevant arts in view of the foregoing description, when read in conjunction with the accompanying drawings and the appended claims. As but some examples, the use of other similar or equivalent operating systems, device types, application types and the like may be attempted by those skilled in the art. However, all such and similar modifications of the teachings of this invention will still fall within the scope of this invention.

Furthermore, some of the features of the present invention could be used to advantage without the corresponding use of other features. As such, the foregoing description

should be considered as merely illustrative of the principles of the present invention, and not in limitation thereof.